Hughes El Segundo Employees Association
Atari Computer Enthusiasts
Reprints of Exchange Newsletter Articles
July 1989 Richard L. Reaser

Do-It-Yourself Calendars

Making Your DeskTop Publisher Work for You!

(2)

By Jim Chapman, S*P*A*C*E

If you are a user of the Timeworks Publisher ST (or similiar) desktop publishing (DTP) software and have a need or desire to create monthly calendars, then this article may be useful.

As the main editor for the Puget Sound Atari News since July 1986 I've been placing a monthly calendar in each issue of this publication. My earliest efforts were easily created with The PrintShop Companion on an 8-bit computer. In July 1988 I began using an Atari ST to take advantage the increased flexability and (laser printed) quality that the ST's DTP software provided. However, there was a price to pay for these advances... Much more time was required to create each calendar! So when it became prudent to switch from Publishing Partner to Publisher ST (to obtain faster printing speed and uniformity with the rest of PSAN - by now being laid-out with Publisher ST), I delayed the calendar portion until I was certain that this change would not increase my already strained monthy workload!

My DTP experiments on the ST indicated that a series of text frames (or columns) should be created - one for each day of the month. Additionally, the date numbers for each of these days must be totally separate entities to avoid precise placement problems caused by 'justification' conflicts. The solution here was to use multiple 'planes' or layers when laying out the calendar. One layer (a single text frame) for the date numbers and another layer comprised of small individual 'day' boxes. And, if this was done correctly, the necessary changes from one month to the next could actually be accomplished with very little effort! (Ahhhh, less work - Isn't that why we got into computing in the first place?)

Creating A Calendar (using Publisher ST - other DTPs might be similiar):

1. Decide on your calendar's general lay-out and exactly what size you want each day's 'box' to be (1" x 1" is good for starters).

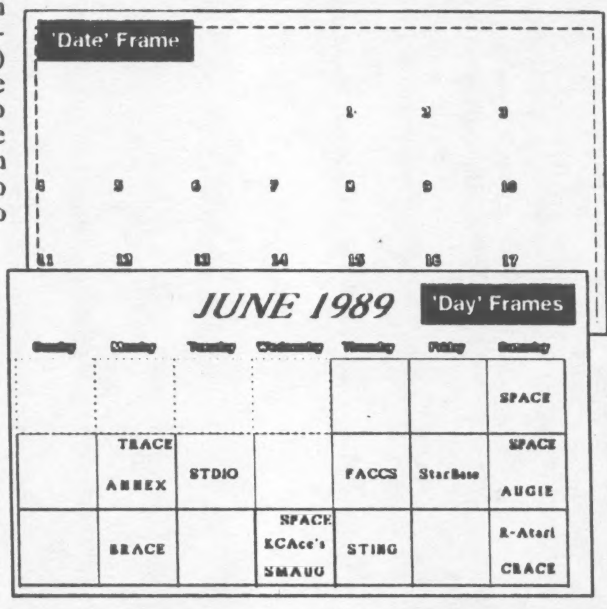
2. In the 'Set Column Guides' dialog box (in the 'OPTIONS' drop down menu):

a. Set the 'number of columns' to 7 (a column for each day of the week).

b. Set the 'GapBetweenColumns' to 0.

c. Set the Left and Right margins as appropriate to allow 7 columns of each daily 'box' (e.g., you'd use 0.75" if each day 'box' was 1" wide and your paper width was 8 1/2").

d. Set top and bottom margins similiarly - making allowances for titles and weekday heading text frames if desired.



3. Click on the 'Frame' tool and draw a 'day' frame (box) one column wide and approximately the desired heigth.

a. Double click on your newly drawn frame to bring up the 'Size and Position' dialog box. Here you can make precise size adjustments as needed. Exit.

b. (In the OPTIONS menu) set Frame Tint to 'clear' and Frame Border to 'All Around' using the thinnest line.

4. Using the Copy and Paste functions of the Edit menu, duplicate the 'day' frame to create six weeks worth (or so) of identical frames. Yes, this is repetitive, but it goes quite fast. **TIP:** From time to time use CTRL C to copy the current frame (or one as close as possible to where you will be working on the page) into the memory buffer. Then simply hit the INSERT key to paste a new frame on the page, and use the mouse to drag it to the desired position. And, if the 'Snap to Guides' feature (in the OPTIONS menu) is ON, your frame will perfectly position itself!

5. Next create a large 'date' frame which completely overlays all of the 'day' frames. (It will later contain the date numbers, 1-31). Initially align the left edge of the 'date' frame with the left column guide and adjust the width so that the frame extends about 1/4" beyond the right-hand edge of the right-most 'day' frames. (This will provide a convenient 'handle' to later access the 'date' frame when making changes for any subsequent month.)

6. Click on the 'Paragraph' tool and create a new paragraph style (I used 'CAL_DATE' as the style name) for the date numbers in the 'date' frame as follows:

a. Justification: Set to 'Table'.

b. Font & Size: Your choice here (I found Dutch, 14 point, Bold, Outlined, to be nice).

c. Dimensions: Set 'Leading', measured in points (72 points = 1.00 inches), to the heigth of each 'day' frame (in our case we use 72 points of leading because the 'day' frame is exactly 1" high). Also, be sure that all 'Indents' & 'Space above' are set to 0.00.

d. Set Tabs: Set tabs 1 through 6 as: 'left tab', 'spaces' leader character, and successively position each tab the width of the 'day' frame from the left border or preceding tab (in our case: #1 - 1.0", #2 - 2.0", #3 - 3.0", etc..).

7. Switch to the Text mode (by clicking on the 'Text' tool) and type the date numbers into 'date' frame as follows: "1 [Tab] 2 [Tab] 3 [Tab]"..... (to the last day of the month - *and no blank spaces*).

8. Switch to the Paragraph mode, click on the numbers just entered (in step 7), and change the paragraph style to the CAL_DATE style (established in step 6). Now You'll see the date numbers expand out to the desired positions. Switch back to the Text mode and position the cursor just in front of the first number. Hit the TAB key a few times and watch all the numbers automatically scroll around to their correct positions. (Nice, huh!) Use the TAB and BACKSPACE keys to move the "1" into the proper day of the week column for the starting month. Exit.

9. Go back to the 'Frame' mode, click on the 'date' frame, and hit ALTERNATE 6 to send this frame to the back

(Continued on next page...)

The NOTEPAD Desk Accessory ③

A HIDDEN WORD PROCESSOR!

A P.D. Review by
Jere W. Frazer
of STDIO and B.R.A.C.E.

The Notepad desk accessory rates an honorable mention in George Terpening's review of the best PD ST software as "a small program that allows you to store a note to yourself." When I read this review I quite agreed. After all, when you click on Notepad in the desk accessory menu, an ordinary looking window opens up. You can type a short note to yourself, like: "Save your work before quitting." or "Today is your wife's birthday. You ought to mention it tonight." Then you can close the window. The next time you open that window, provided you don't turn the computer off, the note will still be there to remind you. Simple and mildly useful. Ho hum. I can probably remember her birthday without going to all that trouble, and I usually save my work without being reminded.

Later on the same day I read George's review, I was looking at the documentation for Dot Magic ST. (Read the review in the March 1989 PSAN and send Chet Walters $10.00 for this program that will print 20+ fonts on an ordinary 9 pin printer. A real bargain!) Anyway, the Dot Magic program includes the Notepad as a desk accessory to be used as a text editor. "To access some special features of Notepad, click again on its name under the Fuji symbol. You can SAVE, LOAD and do other things."

Huh? Are we talking about the same Notepad? I booted up the relevant disk and clicked on Notepad in the Desk Accessories column. The same ordinary window named "Notepad" appeared. I took the mouse cursor back up to the DESK menu and clicked on Notepad again. This time I got a drop down menu that said "COPY, MOVE, DELETE, FIND ..., LOAD ..., SAVE..., PRINT, CLEAR."

This is beginning to look better than a "jot down a note to yourself program." When LOAD is clicked on, the Item Selector appears. I loaded up an ASCII file of over 73k and scrolled through it using the arrow keys, so there does not seem to be any size limitation. If you click LOAD while you have a text file loaded, you then get a choice of REPLACE, APPEND or CANCEL. CLEAR removes all text from the box after an Alert Box warns you of this.

Type some words in the empty box. The Tab Key moves the text cursor five spaces. The mouse arrow can locate the text cursor anywhere you have already put text. You can see that Notepad has wordwrap. The window can be made any size up to full screen. None of the words are hidden if the box is made smaller, but you have to do a lot more scrolling using the Up and Down arrow keys.

When you click on Save, you also get the Item Selector. You can save your text under any name like README.1ST, HISCORES.TXT or NOTEPAD.DOC. These files can be read from the ST Desktop or from Notepad. You have probably gotten my message by now that this is a complete word processor. It really is. FIND is short for FIND/CHANGE, a global search and replace function. I have not yet figured out how DELETE, COPY and MOVE work, but the Delete and Backspace keys work perfectly. New text is inserted rather than overwritten, so editing is simple and easy. PRINT works perfectly. What You See Is What You Get (provided the Notepad window is full screen width); if your window is smaller the program still prints out an 80 column line with wordwrap, but it isn't WYSIWYG. To use Notepad as a text editor for Dot Magic it is necessary to put a Carriage Return after each line.

Notepad is a complete word processor disguised as a one shot memo program. It will create, load, save and print ASCII files accessible from the Desktop as well as the Notepad itself. The price is right: FREE! Since it is a Desk Accessory, it is more conveniently available than a WP program, and will be useful for creating README files or recording scores for games that don't keep track of high scores. You can write letters when you want a rest from your ST application, without having to close it down. You can write PSAN reviews. I plan to use the Notepad a lot now that I know how much it can do. As a matter of fact, I wrote this entire review on the Notepad. Now that the secret is out, I think it should get a higher rating than Honorable Mention.

And now, I have three questions.

1. Who wrote it? There isn't any clue.

2. Why was it disguised? Surely the programmer could have put a hint somewhere about how to access the hidden menu.

3. Are there any other lacklustre Desk Accessories that will take on a brilliant new life if clicked on for a second time?

****

Do-It-Yourself Calendars (cont...)

(behind the 'day' frames). Now you'll be able to see how the date numbers are positioned relative to the 'day' frames. Some adjustment is probably necessary. If so, double click on the 'date' frame (use the 'handle' on the right side) to bring up the 'Size and Position' dialog box. By trail and error, reposition the 'date' frame as desired.

10. Now go to the 'unused' 'day' frames and set their Frame Border to 'none' (or 'send to back'). This makes them invisible.

11. Place text and graphics as desired into the various 'day' frames. Also add other desired title, weekday heading, and graphics frames and text at this time.

Well, that should pretty much complete your basic calendar. Use your imagination to add other features and embellishments. (For example, on the PSAN calendar I've combined two months together with the second month being offset by 0.10 inches and the tint of each 'day' frame set to the lightest shade of fill.)

TIP: If you create your calendar on a 'Master Page', then each time you 'add' a new page (i.e., create a new month), you'll only have to make minor adjustments. With this technique it's easy to do a whole year at a single sitting!

****

S*P*A*C*E
P.O. Box 110576
Tacoma, WA 98411-0576

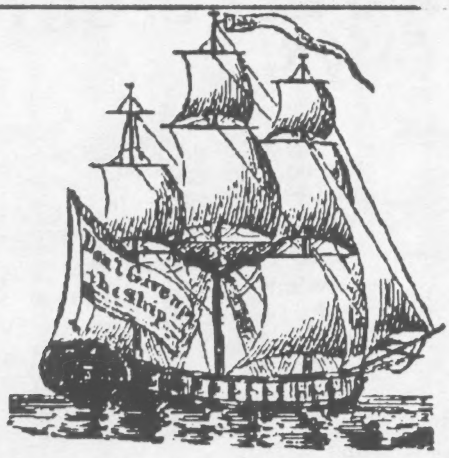
Address Correction Requested

*** DATED MATERIAL - *PLEASE RUSH!* ***

S.B.A.C.E.
Attn: Editor
P.O. Box 83668
LOS ANGELAS, CA 90083

BULK RATE
US Postage
PAID
Tacoma, WA
Permit No. 776

(4)



JUNE 1989
PSAN Atari Activities

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3 **SPACE** Main Meeting, 6pm at Highline Comm. College, Kent
4 	5 **TRACE** Meeting, 7:30pm Thompson Centre	6 **STDIO** Meeting, 7pm Data IO, Redmond	7	8 **FACCS** Meeting, 7pm (Location TBA)	9 **StarBase** Meeting, 6pm Mountlake Terrace Library	10 **SPACE** Tacoma Meeting, 10am, South End Neighborhood Ctr. **AUGIE** Meeting, 2pm, E.BaselineLibrary
11	12 **BRACE** Meeting, 7pm at Lake Hills Library, Bellevue	13	14 **SPACEST** Sig, 6pm, Butler's **KCAce's** 7pm,USRecruiting Offices, Silverdale **SMAUG** 7:30pm	15 **STING** Meeting, 7pm Moreland's Restaurant, Spokane	16 **PSAN June Input Deadline!** *Write about your Atari experiences!*	17 **R-Atari** Meeting, 7pm, Harbor Tower, Oak Harbor **CRACE** 12pm
18  Father's Day	19 **TRACE** Meeting, 7:30pm Thompson Centre	20 **STDIO** Meeting, 7pm Data IO, Redmond	21  Summer Begins	22	23	24 **SPACE** Hardware SIG, 10am, South End Neighborhood Ctr. **AUGIE** Meeting, 2pm, E.BaselineLibrary
25 **FACCS** Board Meeting, Spokane	26 **BRACE** Meeting, 7pm at Lake Hills Library, Bellevue	27	28 **SPACE** **ST SIG** Meeting, 6pm, Butler's, Federal Way	29	30 	1 **SPACE** Main Meeting, 6pm at Highline Comm. College, Kent
2	3 **TRACE** Meeting, 7:30pm Thompson Centre	4 **STDIO** Meeting, 7pm Data IO, Redmond	5	6	7	8 **SPACE** Tacoma Meeting, 10am, South End Neighborhood Ctr. **AUGIE** Meeting, 2pm, E.BaselineLibrary
9	10 **BRACE** Meeting, 7pm at Lake Hills Library, Bellevue	11	12 **SPACEST** Sig, 6pm, Butler's **KCAce's** 7pm,USRecruiting Offices, Silverdale **SMAUG** 7:30pm	13 **FACCS** Meeting, 7pm (Location TBA)	14 August Deadline! **StarBase** Meeting, 6pm Mountlake Terrace Library	15 **R-Atari** Meeting, 7pm, Harbor Tower, Oak Harbor **CRACE** 12pm

JULY '89

The Seattle 'World Of Atari' Show coming this Fall - *DON'T MISS IT!*

(5)

DISK FORMATS

by Malcolm Kay

The term 'disk format' really covers two distinct subjects. At a base level it refers to the manner in which information is physically stored on the disk and at a higher level to the way in which the information is organised to form a file system. In most computer systems the base level is the responsibility of the Basic Input Output System (BIOS) in the operating system, while the higher level functions are handled by the Basic Disk Operating System (BDOS), in the case of the ST known as GEMDOS.

At the base level a number of quite different organisations have been employed, and in particular the Commodore and Apple companies have used uncommon structures but today most small computers, including the ST, IBM-PC compatibles and nearly all CP/M systems, use some variation on an early IBM standard supported by most off- the-shelf disk controller circuits. Common to all such variations is the organisation of the data on disk tracks into a number of sectors each preceded by a header containing the track number, sector number, side number and the size of the sector data area (128, 256, 512 or 1024 bytes). Headers and data areas each begin with special marks identifiable by the disk controller and are each terminated by a two byte check code or CRC. In normal operation the header areas remain unchanged and only the data areas are re-written when the data is altered. In order to allow for some speed variation when re-writing sector data and to allow synchronisation during reading it is necessary to provide gaps between headers and data and between the previous sector data and the next header. The gaps begin with an arbitrary code sequence but end with a sequence enhancing synchronisation during read. Within this structure a number of factors may vary:-

1. The manner in which individual bits are encoded FM, MFM or MMFM. The FM coding is single density whereas MFM and MMFM are double density codes. Fortunately the MMFM is fairly rare.

2. The gap size between the data of one sector and the header of the next. The other gap from header to data is fixed.

3. The data size of a sector.

4. The number of sectors per track and the number of tracks pe side.

5. The number of active sides; one or two.

6. The track and sector numbering. It is usual for the outermost track to be numbered as zero and the numbers to increase sequentially toward the centre. The most common sector numbering is that used in the ST with the lowest numbered sector on each track on each side numbered one, but variations include the lowest numbered sector being zero and the use of different sector numbers on each side of a two sided disk. Sides are not always numbered in the header or this position may sometimes be used to indicate whether or not the disk is two sided.

DISK FORMATS

7. Sector order within a track is not always sequential. The order may be 'interleaved' or 'skewed' with a view to optimising the time for disk access. If it takes sometime for the information read from one sector to be digested then the disk may have moved beyond the start of the following sector before the system is ready to read it. Without interleaving the system must then wait nearly a full disk revolution to read the next sector. In general the controller waits for the correctly numbered sector and consequently the system software does not need to know whether physical interleaving is employed, the only effect will be in respect to disk access times. It should however be noted that some systems (particularly CP/M) employ logical skewing in which the physical sector numbers are sequential but translated via a table to skewed logical sector numbers and in this case a knowledge of the skewing table is essential.

Within the above basic format there remains considerable scope for the organisation of a file system. MS-DOS, PC-DOS and GEMDOS use a common organisational structure in which sequential sectors are logically grouped into equal sized clusters for the purpose of allocation. Items (files) are tabulated by name on a particular part of the disk known as the 'directory' together with their size, creation date, attributes and the number of the data cluster at which the item begins. (For some reason lost in the annals of history and probably related to CP/M, the first available data cluster is numbered two.) After the first cluster pertaining to the item subsequent clusters are found by looking up the entry corresponding to the previous cluster in a File Allocation Table (FAT). Termination of the item is indicated in the FAT by an entry of -1 and unused clusters available for new items by an entry of zero. Clearly without the FAT any information on the disk is completely useless and consequently for security two copies are usually maintained of this table. In the event of loss of the directory it is possible to completely reconstruct each sequence of clusters representing each individual item from the FAT alone although the item names, creation dates, attributes and exact lengths are lost forever.

In order to maintain a maximum flexibility the parameters defining a particular variation within the above possibilities are tabulated on the first sector of the disk, the 'boot' sector. However few if any systems will accept all possible variations of parameters - the ST accepts as many as most. These parameters include:-

1. Sector size - The ST is not consistent here. Sometimes it assumes 512 bytes and at other times it looks to the coded value.

2. Sectors per cluster - The ST norm is two but seems to accept one without problems. Sizes greater than two give erratic results. MS-DOS mostly uses one on single sided disks and two on double sided.

3. Reserved sectors - The number of sectors at the beginning of the disk not available to the filing system. This effectively determines the start of the first FAT. The ST expects and insists on one (for the boot sector).

4. Number of FATs - It seems that all systems insist that this is two. The second FAT follows immediately after the first largely nullifying any added security.

5. Number of directory entries - The root directory has a fixed size and should be a whole number of sectors. With 512 byte sectors there are 16 entries per sector. The

ST norm is 112 corresponding to 7 sectors. I expect the ST will accept up to 256 but have not proven this. The directory follows directly after the second FAT.

6. Disk size in sectors - Number of sides times tracks per side times sectors per track; i.e. the total disk size in sectors.

7. Media byte - A code denoting the format of the disk. The ST codes F8 for single sided and F9 for double sided, but the format does not quite match F8 or F9 formats in MS-DOS. The same code should appear in the first byte of each FAT but the ST format facility always codes F9 in these positions. These disks will always confuse MS-DOS. However the ST does not use this byte and in general has no trouble reading MS-DOS formats provided all parameters are set in the boot sector.

8. FAT size in sectors - Must be large enough to reference all clusters on the disk. The standard ST format uses 5 for both single and double sided disks whereas 2 is sufficient for single sided and 3 for double sided.

9. Sectors per track - This is normally the maximum number that will fit; but opinions differ as to what will fit. The nominal track length on the ST drives in double density is 6250 bytes. A 512 byte sector together with its header and standard gaps occupies 614 bytes or a total of 6140 if we try to fit 10 sectors. To this is added a 60 byte gap at the beginning of the track for a total of 6200 bytes leaving a margin of 50 bytes or 0.8% of the total track length. During formatting the disk controller begins writing a track when the physical index mark is detected and stops when it is again detected so that any format data beyond this latter point is lost. Thus if the disk spins more than 0.8% faster than nominal (still within tolerance) the format sequence will be incomplete. The sequence actually ends with a 40 byte gap and since the starting gap will follow its loss is not too serious allowing a new tolerance of 1.44% in speed, again an allowable disk speed variation. Thus responsible manufacturers have no alternative but to work with a maximum of 9 sectors per track.

10. Number of sides - one or two for a floppy.

11. Hidden sectors - Not often used and must be zero for the ST.

It will be noted that the number of tracks is not specifically stated. It is derived from the disk size in sectors in conjunction with the sectors per track and the number of sides. The innermost usable track is fundamentally determined by the resolution of the magnetic medium and drive heads; as the the tracks become smaller the data is more crammed until eventually it can no longer be reliably decoded. But for the user there are other limitations, how far the heads will actually move on his particular drive, the size of the head access window and the extent of the polished area on the disk surface. Moving inside the standard 80th track can significantly decrease reliability, prevent the disk being read on other machines and above all may dramatically increase head wear by moving off the polished surface of the disk. In my view the meagre gain of 2.5% in disk capacity obtained by employing 82 tracks does not compensate for the risk of lost data or the potential for increased head wear.

ALTERNATE FORMATS There is no valid reason for the oversized FAT tables provided

DISK FORMATS

by the ST format function and it therefore seems prudent to conserve this space even though the difference is rather small.

The use of 10 sectors per track appears to have potential for problems only during formatting. Once successfully formatted data written to the disk should be just as secure as that written to a conventional 9 sector per track disk. So if you find that you can format 10 sector tracks with your drive without squeezing the gaps between sectors then there appears no reason why the 11% or so gain in disk capacity should not be used. On the other hand failure of particular hardware to produce 10 sectors per track cannot be viewed as a defect in the equipment.

Using extra tracks beyond the normal 80 track limit reduces reliability and may increase head wear rate. The use of extra tracks should be discouraged. For disks containing a large number of small files the total space unused in the last cluster of each file can be significant. With a cluster size of two sectors each file may be allocated as much as 1023 bytes in excess of that needed by the file. Setting the cluster size to one sector limits the maximum excess to 511 bytes and may make an extra 50k bytes or so of disk capacity available. The penalties are marginally slower disk access with larger files as the FAT tables must be interrogated more often and larger FAT tables.

The standard interleave factor is one, meaning that sectors are in sequential order. For files containing programs or a mixture of random file types this is probably near optimum on the ST. However in some circumstances, where the major part of the disk is used by some particular types of data files, data retrieval times might be improved by using other interleave factors. It can therefore be advantageous to format with different interleave factors according to the intended application of the disk; but it takes an inordinate amount of time and effort to determine the optimum for each application.

PC COMPATIBILITY Here we are considering only the ability to transfer files literally between machines. There is no practical way to convert executable code in a .COM or .EXE MS-DOS file so that it may be directly executed on the Atari. Data files will usually be found compatible when used with MS-DOS and TOS versions of the same software. Standard text (ASCII) files will not give major problems but MS-DOS software often terminates these with control-Z while indicating the entire allocated size (instead of the bytes actually used) in the directory so that some ST software will 'see' any rubbish in the file beyond the control-Z terminator.

MS-DOS formats of 9 sectors per track will mostly have the critical parameters entered in the boot sector and therefore be correctly interpreted by the ST. But of course there must be hardware compatibility - the ST cannot read 40 track 5.25 inch disks unless it has a 40 track 5.25 inch drive fitted. (A software patch will allow them to be read or written on an 80 track 5.25 inch drive by double stepping.)

It seems that MS-DOS machines depend largely on the media byte at the beginning of the FAT table to identify the disk format. Earlier versions of MS-DOS recognise only a limited range of media bytes unless an extended disk driver is installed, and in any case the careless handling of the media byte by the Atari format utility makes reading these disks on a PC extremely difficult. The easiest way to transfer data on disk from the Atari to a PC, assuming hardware compatibility, is to first format the disk on the PC rather than the Atari. But this is often inconvenient as the PC may not be at hand when the disk needs to be formatted. The alternative is to use a format program on the Atari that

produces MS-DOS formats and in particular pays attention to the media bytes. Such a program is quite straight forward to write and the 40 track formats for both single and double sided disks are well standardised. For 80 track disks the formats recognised by various MS-DOS systems is rather vague, and some experimentation may be necessary to find a format recognised by a particular MS-DOS system.

Many MS-DOS systems now incorporate an 80 track high density drive of either 5.25 inches or 3.5 inches. The connection of a high density drive to the ST cannot be done through the external floppy drive connector as the built in disk controller of the ST cannot handle the higher data rate. However a high density drive with a Small Computer Systems Interface (SCSI) might be matched up to the hard drive port on the ST but would also require a software patch to the operating system. At least for the moment SCSI drives are expensive and the approach is therefore unattractive.

UNIVERSAL FORMAT PROGRAM Disk formatting is the operation of creating the sectors on the disk and initialising the disk for the filing system by setting the parameters in the boot sector and zeroing the FAT and directory areas.

I have written a format program for the ST which allows access to useful (and perhaps some less useful) parameters. The program ensures that the selected media byte appears in all three positions (boot sector and each FAT) and allows viewing and copying the format from another disk. The program is intended for those who wish to experiment with formats rather than as an effortless way to repeatedly reproduce a particular format. Formatting is verified track by track and is therefore significantly slower than programs which do not verify. But who would want to save a few seconds during formatting at the risk of a disk failure at some critical point during use? The program will be made available to the club public domain library.

Tentatively planned extensions to this program include:-

1. The ability to format a 5.25 inch disk with 40 tracks at 48 tracks per inch by double stepping in an 80 track drive of 96 tracks per inch.

2. The ability to install an MS-DOS system.

3. The ability to install an executable TOS boot sector.

A BRIEF LETTER TO THE EDITOR(S)

DEAR SIR(S)

Please find listed below my strengths and weaknesses...
They may help you to better plan magazine content and club direction:

Software I use regularly now	Expertise Rating (0→10)	Purpose

Software I would need help with or am curious about...

NAME _____ Phone _____

PLEASE COMPLETE AND HAND TO JAMIE

"WHATIS" FILE IDENTIFIER

by Bill Aycock

WHATIS File Identifier, v1.6
(c)1989 by Bill Aycock

WHATIS is a simple utility that will identify 23 different types of files.

Running the program is very simple--just binary load the program from any DOS. When you're asked for the name of the file to identify, type in its name. If you don't include a device specification, WHATIS will add D: to the filename.

WHATIS will then read the first few bytes of the file in question. If these bytes match a known file "signature", WHATIS will tell you what type of file it is (or will say "TEXT (or data)" if the file doesn't match any of the known types. The program will then wait for you to press the START key before returning to DOS.

SpartaDOS users: you have the option of passing the filename on the command line if desired. Also, you won't be prompted for the START key, since Sparta doesn't clear the screen when entering the command processor.

So far, WHATIS can recognize files prepared with these compression utilities:

- ARC
- ALFCRUNCH
- CRUSH
- DISKCOM (Disk Communicator)
- MASH
- SCRUNCH
- SHRINK

Decoders for these file types are available in LIB 3 of ATARI8 on CompuServe and elsewhere. In addition, WHATIS will recognize these types of files:

- SAVEd BASIC programs
- EXTENDed BXE (BASIC-XE) programs
- SAVEd MAC/65 code
- OBJECT code (machine language)
- DaisyDot fonts
- GIF pictures
- compressed KOALA pictures
- SpartaDOS X (SDX) external commands

as well as these types of ST files:

- executable ST programs
- ST-Writer files
- Degas pictures (.PI?)
- Cyber (.SEQ) animations
- Spectrum pics (.SPC and .SPU)

It _is_ possible to fool WHATIS. If a data file happens to start with the same bytes as one of these file types, it will be identified incorrectly. Also, SpartaDOS X device handlers will usually show up as DISKCOM files.

WHATIS was based on Roy Goldman's Compactor Detector, a BASIC program which identifies files and allows renaming them to have a "standard" extender. WHATIS was written in Action! and compiled with the RunTime Library, both of which are available from the fine folks at ICD. Some of the I/O routines used were written by Don Davis.

I'm always looking for more file types to support in WHATIS. If you know of a particular type of file that always start with the same few bytes, please let me know--I'll be glad to add it in!

(Editor's note: WHATIS.COM can be a boon to BBS SysOps and Users alike. Many are the times when I have no idea what kind of file (or kind of compression routine) I received late last night. WHATIS and the Compactor Detector saves me much time and many keystrokes determining what this file really is.)

MILE-HI
6-89

INTRODUCING MAX SYSTEMS

by Marty Albert

Quality Software for Atari and
Commodore 8-Bit Computers

This document is for information only and in no way states or implies any contract. It may be freely distributed as long as it remains unchanged. Reformatting for printing is permitted.

Copyright 1988 by MAX Systems

MAX Systems
Suite 6-216
4005 Manzanita Ave.
Carmicael, CA
95609-4005

Thank you for taking the time to read this introduction to MAX Systems! We think that you'll find the information here both interesting and, possibly, profitable!

In recent years, the software available for the Atari and Commodore 8-bit computers has more or less "dried up". There is little new commercial software being released by the big publishers and the quality of Public Domain software has been steadily decreasing, with a few notable exceptions. The support for PD and so-called "shareware" has been sparce to say the least. Often, when you get a PD or shareware program, you can't find out who to ask when you have a question because the author is not available or has even moved on to other computers.

program, often with poor results.

What about the other side of the coin? You have written a program that you feel is a good one. You send it off to a few publishers, but they reply that, "...we don't support that old machine anymore." You are left with three alternatives:

1) Release it as PD and make no profit from it.
2) Release it as "shareware" and make little, if any, profit from it.
3) Just keep it and the heck with it all, wasting your time and effort that you put into the program.

This hurts everyone! The non-programmers see a lack of good software for their machines because the programmers are not seeing any rewards for their work and stop supporting the machine. The programmers get disgusted and stop writing new software because they are going broke. As time passes, it just keeps getting worse!

That is where MAX Systems comes in!

Just What is MAX Systems?

MAX Systems believes in the FIRST Ataris and Commodores! They are fine machines, capable of amazing feats. They are far from dead. We want to help both the programmer and the user by providing a way to get quality software from the programmer to the user while keeping cost to the user low and profit to the programmer high.

A typical software package, complete with documentation, would be priced at about $20.00 with about $10.00 going to the author. MAX takes care of the details!

How Can MAX Do That?

Simple! MAX uses "low tech" systems to duplicate the disk and documents. There is no big overhead so MAX can keep costs way down. We pass this savings on to the users and still return a large percent to the programmer.

All programs are sent on high quality diskettes with clear, easy to read documentation. All diskettes are guaranteed for life against failure. Update policies vary depending on the program in question and are fully documented with the software.

MAX Systems DOES NOT use copy protection. Copy protection simply adds to the cost. If someone really wants to copy the program, they will do so, no matter what you do to the disk. Instead, the programs are very document dependant. Someone may copy a disk, but they won't sit down and type in 40 pages of docs!

MAX uses "no frills" packaging. Sure, that four color slick paper box looks nice, but does it make the program run better? No, it doesn't. So, why add that cost? We at MAX see no reason for fancy packages.

dealer network. Dealer orders are, of course, welcome, but they are treated just like any other order, with the exception of pricing for large quantities. Most software packages have a quantity discount that is available to anyone that wants to order in larger supplies, such as a User Group or dealer.

MAX advertises by modem. By the use of the national online data services and BBSes throughout the country, MAX reaches the people who need software. Why pay $10,000.00 or more for an ad in a magazine when as many people can be reached by modem for less than $5.00?

Isn't MAX Just Another PD Place?

There are many companies now in the business of "selling" PD software. For $5-$10, you get a disk of useless programs with maybe one good one mixed in. No documentation, no support. Not only do we at MAX feel that this is NOT the way to get quality software, we also question if it is even legal due to copyright laws, but we are not lawyers.

Instead, the programs offered by MAX Systems are ALL copyrighted programs. Each one is registered with the United States Copyright Office. The author retains ownership of the program and grants MAX the permission to distribute the program. This provides the author and MAX a legal recourse against anyone who "pirates" the software.

As you can see, we are far from just another PD software disk maker!

What Benefits to Programmers Does MAX Offer?

The benefits are many. First off, your program is REALLY copyrighted! It is registered with the US government!

Next, MAX handles the details like disk duplication, printing of docs, shipping, order taking, and other "paper shuffling". That leaves you free to do what you do best--program.

MAX also deals with the money side. Each quarter, MAX will send you a royalty statement and check based on the sales of your program(s). To guarantee your royalties, each time a copy of your program is sold, MAX places your royalty payment in a special bank account. Quarterly payments are made from this account to you. MAX deals with bad checks, etc. from users so you don't have to.

The bottom line? Just sit back and enjoy the extra income four times a year!

What Does it Cost Me to Let MAX Sell My Program?

Nothing! All that MAX requires from you is that you provide the program and COMPLETE documentation in a standard, non-protected disk file format for the machine that you have written it for.

MAX may, from time to time, contact you with technical questions from users. MAX may also ask for new versions and/or updates as needed.

MAX does reserve the right to edit the documentation for clarity, but you will always be asked for approval of the revised docs. It is, after all, your program!

What About Credit for the Program?

You are encouraged to have your name appear on the title screen of the program. We want the users to know that you wrote it!

MAX also requires that "Distributed by MAX Systems" appear somewhere in the program for the users to see.

We encourage you to refer questions about the program to MAX. This way, you won't be bothered except for real problems.

What About Questions?

Many programmers rely on questions from users as feedback for future revisions. MAX understands this! We will maintain a file of questions, comments, and problems from users and will be happy to provide this to the programmers.

What Machines Does MAX Support?

At this time, MAX supports the following computers:

Atari

400 800

600XL 800XL 1200XL

130XE XEGS

Commodore

64 (all versions)

128 128D

We will be adding other machines in the future such as the Atari ST, Amiga, and IBM.

We are, as always, open to comments and suggestions for what machines to support, so please feel free to drop us a note!

What Sort of Programs is MAX Looking For?

12

Just about anything! Games, utilities, applications, whatever!

In the Commodore area, there seems to be a lack of serious applications software right now, but games are always popular.

For the Atari, again games are always good, but there too is a lack of good applications.

The language that the program was written in is not important. BASIC, compiled BASIC, C, assembler, Action!, Pascal, whatever. Remember that in order to copyright your program, MAX must have the COMPLETE source code listing no matter what the language used!

What Does MAX Offer the Users?

In one word, support. Not only are they getting a piece of quality software with complete documents, but they know where they can reach MAX Systems if they have a problem or question.

The user will also get a sense of support from themselves, knowing that they are doing something to help their own machine by purchasing a product written by an independent programmer.

So, How Do I Get More Information?

This is just an introduction to MAX Systems. If you are a programmer and are interested in more information about MAX and how to submit a program to us, please send a SELF ADDRESSED STAMPED ENVELOPE to:

MAX Systems
Suite 6-216
4005 Manzanita Ave.
Carmicael, CA
95609-4005
ATTN: Marty Albert

MAX can also be reached on GEnie at address:

MARTY.A

Please be sure to tell us what machine that you are interested in programming!

Thanks again!

PROGRAMMING FOR FUN

Exploring 'Modular Arithmetic'!

ST Tutorial by Brian Hogan, S*P*A*C*E

13

If you hadn't already guessed before you read my "bio", I'm one of those oddballs called a math teacher. Hence I'm always on the lookout for interesting math topics as it relates to graphics. Sometimes it doesn't have to be a real exotic concept to get some impressive results.

A useful, but little known math operation, is the MOD operation. MOD refers to modular arithmetic and modular arithmetic refers to using the remainder when division is performed. x MOD y give the remainder when x is divided by y. It generally is used when x and y are positive integers. In that case the answer will be and integer 0 through y-1 Thus 5 MOD 2 = 1; 22 MOD 13 = 9; 3 MOD 5 = 3; 22 MOD 3 = 1; 22 MOD 2 = 0.

Every once in a while I forget myself and do try to write programs that are not graphics oriented. Being a math teacher, I sometimes try to write that perfect CAI (Computer Assisted Instruction) program for my algebra classes. Since fractions are always the bane of students I recall trying to write a program that went into the process of finding the least common denominators for adding fractions (The LCM) and finding the greatest common divisor (GCD) for reducing fractions. To aid in these tasks is an algorithm as old as the hills called the EUCLIDEAN ALGORITHM. The algorithm finds the GCD of two numbers. Let me illustrate it by an example, then write a short program to do the same thing.

Suppose we wish to find the Greatest Common Divisor of the numbers 18 and 30.

Step 1: Divide 30 by 18, keep the remainder (which is 12)

Step 2: Divide 18 by 12, keep the remainder (which is 6)

Step 3: Divide 12 by 6, which gives a remainder of 0.

Step 4: Since the remainder was 0, STOP.

The GCD is the last divisor, namely 6, in this case.

To summarize, keep dividing the previous divisor by the remainder until the remainder is zero. Then the GCD is the divisor used that gave us the remainder of zero.

As you can see, the remainders are the vehicle of the algorithm, hence the MOD function should be able to help us to write a subroutine to do the job we want. As a side benefit, I'll demonstrate passing a variable to a subroutine by using the VAR keyword in the definition of the gcd procedure. This is new to the version 3 of GFA BASIC.

```
' A GCD demo
DO
  INPUT "Enter two positive integers",a,b
  gcd(a,b,g)     ! values a and b are passed,
                 ! g returned
  PRINT a,b,g
LOOP
'
PROCEDURE gcd(x,y,VAR r)
                 ! parameters are local
  IF x<y THEN
    SWAP x,y     ! Make divisor the
                 ! smaller number
  ENDIF
  REPEAT
    r=x MOD y    ! remainder (it will be
                 ! smaller than y)
    x=y  ! Set it up so we divide into the
    y=r  ! previous divisor by the remainder
  UNTIL r=0
  r=x    ! previous divisor>> pass
         ! back to variable g
RETURN
```

Just for fun, I'll show you a "more primitive" subroutine that only involves repeated subtraction. Since division is repeated subtraction, you might suspect that the two different procedures might be cousins of each other. You might do a benchmark to see which routine is faster (some people like to do that stuff!).

```
' A GCD demo
' An alternate way, no MOD used
DO
  INPUT "Enter two positive integers",a,b
  gcd(a,b,g)
  PRINT a,b,g
LOOP
'
  PROCEDURE gcd(x,y,VAR r)
    WHILE x<>y
      IF x>y THEN
        x=x-y
      ELSE
        y=y-x
      ENDIF
    WEND
    r=x
RETURN
```

So now you can reduce fractions, just send the numerator and denominator through the GCD process, then divide the numerator and the denominator by the GCD to get your new fraction. Ok, what about adding two fractions? Well we need the LCM (The LEAST COMMON MULTIPLE) of the two denominators (ie. we need the smallest number that the two denominators will evenly divide into).

It turns out that if the product of the two denominators is divided by the GCD of the two denominators, you'll get the LCM. (Is your math good enough to prove that? -- not too easy, by the way). Thus if the denominators are 36 and 24, multiply 36 by 24 (get 864), find the GCD of 36 and 24 (get 12), divide 864 by 12 to get 72. 72 is the LCM, hence the least common denominator.

Ok, I'm sure you'll all go out and try your hand of writing a program for your kids so they can practice adding fractions. But before you do, you might try this next little "for fun" program. This program shows some pictures as a result of using modular arithmetic. I think you'll find them quite pleasing.

The program allows you to first choose a value of N (some positive integer). Try a fairly large value for the first time, maybe 96. Now think of a circle being divided up into 96 equal parts and numbered 0 through 95.

Now a multiplier P is chosen (try 2 or 3 or 94 or 48...). Now, via a loop, the product of a*P is calculated, for values of a being 0, 1, 2, 3, ... 95.

The results of each product is then reduced to an answer between 0 and 95 by the MOD operation--call this number b. (Thus if N=96 and P=4, for a=30, a*P MOD N would result in the number 24 -- call this number b.) The value of "a" and the result of the modular arithmetic, "b", is then graphed on the circle by connecting the point "a" to the point "b" (remember the circle is numbered 0 through 95). For our little example above, you would connect the point labeled 30 to the point labeled 24.

That's it -- try out the program. To completely understand how it finds the points on the circle requires a little knowledge of trigonometry. The CIRCLE command always draws a perfectly round circle, taking the aspect ratio into consideration. Finding a particular point on the circle required a little experimental trial and error, which accounts for some of the weird values in the trig functions in the graphline procedure.

```
' A modular arithmetic demo
' Run in MEDIUM Resolution
' Initialize some variables
GET 0,0,500,150,erasecircle$
                        ! To erase the circle
sp$=SPACE$(60)     ! To erase text
SETCOLOR 3,7,7,7 ! Set text to white
SETCOLOR 0,0,0,0 ! Color screen black
COLOR 3
CIRCLE 320,100,150
DO
  PRINT AT(1,1);sp$
  PRINT AT(1,1);
  INPUT "ENTER N(b=a*P MOD N)
        (0 to quit) ",n
  EXIT IF n=0
  DO
    PRINT AT(1,2);sp$
    PRINT AT(1,2);
    INPUT "ENTER P(b=a*P MOD N)
          (0 will choose new N) ",p
    EXIT IF p=0
    PUT 40,30,erasecircle$
    COLOR 3
    CIRCLE 320,100,150
    FOR a=0 TO n-1
      b=(a*p+q) MOD n
      COLOR 2
      graphline(a,b)
    NEXT a
  LOOP
  PRINT AT(1,2);sp$
LOOP
PROCEDURE graphline(a,b)
  LOCAL x1,y1,x2,y2
  x1=148*COS(2*PI*a/n)+320
  y1=66*SIN(2*PI*a/n)+100
  x2=148*COS(2*PI*b/n)+320
  y2=66*SIN(2*PI*b/n)+100
  LINE x1,y1,x2,y2
RETURN
```

Editor's Note: Brian Hogan, the author, is a long time Atarian (both 8-bit and ST) and avid BASIC language programmer. He is employed as a mathematics and computer science instructor at Highline Community College in Kent, Washington. This article is the eighth in an ongoing series of BASIC tutorials for novice programmers that he has written for PSAN. Your comments (both pro and con) as well as any suggestions for future topics which you'd like Mr. Hogan to discuss herein are encouraged and appreciated.

****

DRAFIX CAD - *One Year Later*

Reviewed by Ken Wildes, STDIO

It was a little more than a year ago I bought my MEGA 2. With it I also purchased a Hewlett-Packard 7475A A/B size plotter and DRAFIX CAD v1.0. Needless to say I was very enthusiastic about it. This enthusiasm was quickly dampened after I hooked up the system and tried to make my first plot. Nothing. Just the error light on the plotter blinking at me like some malevolent eye. The wife and I both went from cheery and bright to dull and gloomy just that quick. After checking all the connections and cables our hopes were up again, and again dashed. The next morning I called Foresight Resource Corp. and was told that "the Atari guy isn't here right now, but if you change parity, blah, blah, blah..." I wasn't too sure. But I did what he told me and was still without results. About this time my wife was very calmly trying to keep me from launching a jihad against Foresight Resources. She finally returned my wits to me, and I wrote them a letter discussing the problem in detail. In about 10 days a package arrived in the mail from Foresight. It contained two brand-new disks and an apologetic letter. They worked! In no time at all my plotter was happily tapping out a drawing. This was good service. When I found out that they were in the middle of a move from Lawrence, Kansas to Kansas City, Missouri I upgraded that to GREAT service.

So what do I think of Drafix now that I've been using it for a year? I think its just fine! I've used both AUTOCAD, and CADKEY on other systems at work, and yes they are better than DRAFIX. However, neither of them have any interest in producing an Atari version, and both cost over $3,000. That's more than my entire system, plotter and all!

Besides the price, which is just under $200, one of the reasons I went with DRAFIX in the first place is the lack of windows in the drawing area. Yes, the lack of windows! I am by nature a visually oriented person. When I'm working on a drawing I have to be able to see it to be able to think about it. Strange, but true. I didn't want a CAD program that would interrupt my visualization with a gigantic box in the middle of it asking me what I wanted to do next. Drafix has one small drop-down window in the upper left corner of the screen which is used for file functions and a few other non-drawing functions. This leaves the rest of my "brain" untampered with. DRAFIX is very easy to learn. You could describe it as an almost linear process. You read about a function in one of the manuals. (Three supplied.) Practice it a few times to get the feel, and then use it.

The menu bars run horizontally across the top of the screen, and are logically ordered. They even use all three buttons of the Atari two-button mouse. (I haven't flipped my wig, read on.) Usually the left button is used to select and implement a function. The right button is used to cancel the function. The "middle" button is pressing both buttons at once to select an alternative of a main function. When I first read about this, I had my doubts, but it seems to work every time.

There are some things I'd like to see implemented in the Atari version of DRAFIX. At first I noticed some little dots beside some of the commands in their manual. I skipped back to the intro. and found out that these are commands not available to the Atari version of DRAFIX. Capabilities such as being able to smooth a set of lines with sharp corners, called "splining". Having all of your personal choices of parameters installed upon boot-up. Another is "sketching", being able to draw free-hand. (Handy when you're "signing" a finished drawing.) Having sent and received letters from Foresight Resources the word from them is that they will update the Atari version of DRAFIX along the same lines as their other versions. But this depends on the market, I'd guess.

All in all I'm satisfied with DRAFIX. It is very good value for the price. No complaints about the service either. Foresight has recently released v1.01 of DRAFIX for the Atari which contains the ability to use DRAFIX drawings made on an IBM, and a "Dot Plotter" which supposedly turns your dot-matrix printer into a high-res. plotter. When I get my upgrade ($35 to those already registered) I'll let you know how it goes.

Ken

****

PREPX
SpartaDos X
Disk Preparer

Dave Arlington - JACG

4-89

No doubt, the premium product release for the Atari 8-bits in 1989 is and will be the SpartaDos X cartridge from ICD, Inc. There will be plenty of words written about it this year, but one of its most powerful capabilities is that you can custom configure the Dos to your system using a file called CONFIG.SYS.

As a startoff point, the SpartaDos X cart itself has a default CONFIG.SYS file burned into its ROM that is installed if you do not provide your own on disk or if you hold down the option key while booting. Since ICD has no way of knowing the system configuration of every purchaser, they took their best shot when deciding what to include or not include in the default CONFIG.SYS file.

Since chances are the default CONFIG.SYS file will not be exactly right for your system, it really makes sense to write the one sector CONFIG.SYS file to each disk you have. There are several reasons you might want to do this. First, every device driver you load that you don't need or want takes up memory. For instance, the default file loads both CLOCK.SYS and JIFFY.SYS even though you only need one or the other depending on whether you have an R-Time 8 cartridge or not. Secondly, it is annoying to continually see messages like "R-Time 8 not present" or "Error while saving memory" for options you don't have present. Lastly, even when the options apply, I question some of the default choices. On a stock 130XE, the extra memory is made available for programming with Basic XE, rather than being used for a RAMdisk. I think most users would prefer the RAMdisk. When a RAMdisk is installed, it is put on Drive 9, inconvenient if all your software using RAMdisks is set up for Drive 8.

After hopefully imparting the wisdom of creating your own custom CONFIG.SYS files, let me explain what PREPX will do for you. To avoid creating your own CONFIG.SYS files by hand or creating one and having to copy it to many disks, PREPX will let you prepare as many disks as you want in one session. You need only press a couple of keys to customize your disks exactly the way you want them. You can use it to format disks for new storage. If you have 2 free sectors on your old disks, PREPX will write the custom CONFIG.SYS and/or AUTOEXEC.BAT files and then hide them so your directories will not change. If you type the program in using the Action! cartridge, you can even change PREPX's defaults to match your own special system.

(15)

Following are the valid keystrokes and a brief explanation of some of SpartaDos X tricks of the trade:

M - Memory Change. Defaults to 48K for Atari 400/800, 128K for XL/XE models. Every time you press M, the memory size changes to tell PREPX how much memory your system has. The options are 48K, 64K, 128K, and Extnd for anything over 128K. Every time you switch from Extnd to 48K or from 64K to 128K, the defaults for RAMdisk, BASIC memory save, and Cartridge memory save will change. The defaults for all 3 are (N)one for 48K and 64K, and Drive 8 for 128K and Extnd.

R - RAMdisk. This allows you to set the drive number or (N)one for any internal RAMdisks. Since this refers to internal RAMdisks and not devices like the MIO board, you will not be allowed to specify a drive number if the memory toggle is set to 48K or 64K. To use the 130XE's extra memory for BASIC XE programming, set memory to 128K and RAMdisk to (N)one. Note, every time you change the RAMdisk setting, the BASIC memory save and Cartridge memory save also change to the same option. To toggle these separate from the RAMdisk, toggle these AFTER setting the RAMdisk option.

B - BASIC memory save and
C - Cartridge memory save. This allows you to set the drive number or (N)one for these memory saves. Unlike internal RAMdisks, these can be saved to floppy drives, hard drives, or MIO RAMdisks, so these two toggles are available in 48K or 64K mode. However, a word of caution before you designate the memory saves to a floppy drive. Any time you change memory AT ALL, SpartaDos X saves the whole chunk of memory to disk. For instance, if you load a 32K BASIC program and then type NEW, SpartaDos X will save the whole 32K block of memory anyway since all it sees is that memory has been changed in some way. So use these options with caution with floppy drives.

A - AtariDos Access. Toggles (Y)es or (N)o. If you want to be able to read and write to or from disks with a standard Atari DOS format (standard meaning DOS 2.0, 2.5, MyDos, SmartDos, Dos XL, but not DOS 3 or DOSXE), leave this option at (Y)es.

S - High Speed. If you own a Happy or Indus disk drive, leave this at (Y)es for high speed read/writes. Actually, since the Indus driver takes up no memory, it doesn't waste memory to leave this at (Y)es even if you don't have an Indus or Happy drive.

T - R-Time 8 Cartridge. Uses CLOCK.SYS driver if set at

(Y)es, DIFFY.SYS if set at (N)o.

H - Hide CONFIG.SYS file. If set at (Y)es, the CONFIG.SYS file and/or AUTOEXEC.BAT files will be hidden after being written. There are two nice things about hidden files under SpartaDos X. First, it acts like protecting these files since you cannot overwrite them accidently when using PREPX. Secondly, you won't have the same two files cluttering up every disk directory like DOS.SYS and DUP.SYS always did with DOS 2.5. The only files that have to be seen on a SpartaDos X disk are your actual data and program files. How nice!

P, D - Prompt and Path Display respectively. Hitting either of these two keys, turns on the cursor in the proper field. You may type in text until you either hit RETURN or reach the end of the edit field area. No real edit checking is done, so what you type is what you'll get. Note the PATH will still always begin with CAR: and edit-checking of the $ symbol in the Prompt is not done for those who want actual $ symbols in their prompt.

X - XEP-80. Toggle to (Y)es if you want XEP-80 support, leave at (N)o otherwise.

The last four options are interrelated in that they cover popular options on boot-up that are handled in an AUTOEXEC.BAT file rather than in CONFIG.SYS:

W - Write AUTOEXEC.BAT. If set at (N)o, defaults change to Key Buffer: Off, Left Margin: 2, Prompt for Time/Date: N and these defaults cannot be changed. Conversely, if set at (Y)es and the above three values are in place for the K, L, and I options, the AUTOEXEC.BAT will not be written.

K - Keyboard Buffer. Toggles On or Off the type-ahead keyboard buffer. The SpartaDos X default is Key Off, unlike SpartaDos 3.2d. Also unlike SpartaDos 3.2d, when the keyboard buffer IS on, the key repeat rate IS NOT speeded up so you can now use the keyboard buffer with Action! for instance. Note the PREPX default is On. This buffer can interfere with some software, so turn it off if you have problems.

L - Left Margin. Toggles to 0 or 2. PREPX default is 0, default without AUTOEXEC.BAT is 2.

I - Initialize Time/Date. If set to (Y)es, you will be prompted to enter time and date on boot-up, like an MS-DOS machine. If you indicated you have an R-Time 8 cartridge, this will be set to (N)o and you will not be allowed to change it.

When you have selected all your options, press START to continue. The ICD SpartaDos X format menu comes up. If you wish to format the disk, enter the relevant information and press F. If you only wish to write the CONFIG.SYS and/or AUTOEXEC.BAT files to a disk that has already been formatted, press ESCape. When done, you will be returned to the PREPX options menu. From here you may set up for another disk or press ESCape to exit the program.

(16)

Behind The Scenes

The program is pretty straightforward. If you wish to set the initial parameters to fit your own needs, you need only make a couple changes to InitVar() and DrawScreen().

CheckOption() reads the key pressed and branches to the proper handling section. WriteOptions() calls the SpartaDos X format routine and then writes the actual files.

For the prompt and path options, I could not get the library procedure InputS() to work properly, hence my own limited GetS() procedure.

The Toggle() procedure might be of interest to beginning Action! programmers since it covers a technique familiar to C programmers, but one that is not mentioned anywhere in the Action! manual. Unlike BASIC, when you pass variables to an Action! procedure (think of a BASIC subroutine), you are really only passing the VALUE of the variable, not the actual variable itself. This is nice when we want to be sure our main variables are not mysteriously altered by some procedure somewhere.

Sometimes though, as with the Toggle() procedure, we do want access to the variable itself so we can change its value. We do this by passing the ADDRESS of the variable (@hide), instead of the value. The procedure being called then uses a pointer of the same type as the varaible. (For instance, in Toggle() we use a BYTE POINTER memloc to hold the address of the variable (a BYTE) being passed.

By using this pointer, we have the same access to the variable we had before, using the ^ operator (as in memloc^, which in this example, is like using the variable hide). The additional benefit is that any changes we make to memloc^ are the same as changing the value of the main variable.

I hope this utility is just the start of a lot of support for the SpartaDos X operating system, and makes customizing your own disks a little easier.

8-BIT PUBLIC DOMAIN PRODUCTIVITY

Examining both SPEEDSCRIPT and SPEEDCALC!

By Don Lucia, AUGIE

Since we have so many new 8-Bit owners in the ATARI community, I thought that a review of some of the more easily learned and available productivity programs would help those new to ATARIDOM.

SpeedScript 3.0
a Review

Need an inexpensive Word Processor? Well, you don't have to look any farther. SpeedScript 3.0 is Public Domain software available in most club libraries. The documentation was printed in the May, 1985 issue of Compute magazine, but most clubs have it on disk. This program requires at least 24K of memory, a disk drive or cassette and an 8-bit Atari (400/800, 600XL/800XL/1200XL, or XL/XE series), and access to a printer.

While this program is very compact in size, it has many of the features found in most commercial word processors. SpeedScript 3.0 is also very easy to learn. You type in everything first; preview and make corrections on the screen; delete words, sentences, and paragraphs; then print out an error free draft, letting SpeedScript take care of things like margins, centering, headers, and footers.

When you run SpeedScript, the line at the top of the screen is the Command line which presents all messages. The other 18 lines are for text. The screen shows 40 columns and uses word wrap so you don't wind up with split words as you do in BASIC. The program uses standard Atari type scrolling, screen formatting and text editing, so you'll feel right at home with this word processor.

Most of the commands however require the CONTROL key as well as the normal key. For example to check for unused available memory, you would hold down the CTRL key while pushing the "U" key. The available memory would be displayed on the command line.

SpeedScript 3.0 supports up to 4 disk drives and has a Mini DOS built in (delete file, lock/unlock, format disk, select drive or load a file). This program will also swap characters when you type them out of sequence; change character from uppercase to lowercase or vice versa; change text luminance; change border color; widen or narrow text to prevent screen loss; show word wrap spaces (false spaces) as tiny dots on your screen; types inverse video characters; as well as screen/print formatting, special characters, fonts, embedded printer controls and many more special features. A nice addition to anyone's library.

!!! MAN HAS 12 BILLION BRAIN CELLS THIS WILL GIVE YOU SOME IDEA OF THE UNEMPLOYMENT SITUATION. !!!

SpeedCalc
a Review

Have you ever wanted to track stocks, bonds, checking accounts,mortgages, rent payments, leases, inventories, etc., etc..? Well, if you have, that is what a spreadsheet is meant to do and SpeedCalc is a super Public Domain spreadsheet available in most club libraries. The documentation was published in the March, 1986 Compute! magazine, but most clubs have it also in their library. This program requires at least 48K of memory, a disk drive, the 400/800/XL/XE Atari machines and access to a printer.

A spreadsheet allows you to enter and organize data, then perform calculations that produce new information. Simply put, a spreadsheet program is an electronic equivalent of the familiar paper (columnar) worksheet. Besides being much faster than its paper counterpart, the electronic spreadsheet allows editing features and figure manipulation of large amounts of data with minimum effort.

The top line, just as in SpeedScript, is the Command line where all messages and instructions appear. Lines 2 thru 4 are the input buffer area. This is where you enter field titles, data, or formulas. The lower 19 screen lines are the window of your spreadsheet and are laid out with the vertical columns identified by the letters AA thru BK, while the horizontal lines are numbered 001 thru 100. The square where the horizontal and vertical columns meet is called a cell. This program has the capacity to use 5,000 cells.

The movement of the cursor and the editing features are the standard Atari keys plus in most cases the CTRL key is added. Three types of data are supported, text, numeric, and formula. Text is not truly data but it may consist of comments, column headings, titles, or remarks. Numeric data are the numbers you wish to enter (up to 36 digits) for record or calculation. Formula data is a mathematical expression. It may be as simple as 2+2 or as complex as your imagination allows. **The formula must always start with the (=) equality sign or the program will either give an error or treat it as text.** The program supports the following mathematical operators:

Operator	Function
+	addition
-	subtraction
*	multiplication
/	division
^	exponentiation
=	equality

Formulas may also include the following Functions:

@ABS ()	absolute value
@AVE ()	average of a block of cells
@EXP ()	natural exponent
@INT ()	integer
@LOG ()	natural logarithm
@RND ()	round to nearest integer
@SGN ()	sign
@SQR ()	square root
@SUM ()	sum of a block of cells
PI	value of pi (3.14159265)

The program is capable of constant or selective recalculation with a toggle

(Continued on Page 25...)

```
3590 X=LEN(PATH2$)
3600 X=X-1
3610 IF PATH2$(X,X)=">" THEN 3650
3620 X=X-1
3630 IF X=0 THEN 210
3640 GOTO 3610
3650 BF$(LEN(BF$)+1)="CWD "
3660 BF$(LEN(BF$)+1)=PATH2$(1,X)
3670 BF$(LEN(BF$)+1)=CHR$(155)
3680 IF CO=0 THEN 1120
3690 AFIL=1:BFIL=15
3700 ADES=1:BDES=15
3710 IF BFIL>LEN(FIL$) THEN 3880
3720 POSITION 14,12
3730 ? "Dup - Check"
3740 FILE1$=FIL$(AFIL,BFIL)
3750 GOSUB 2020
3760 IF AFIL=ADES THEN 3820
3770 POSITION 1,12
3780 ? FILE1$(3,14)
3790 POSITION 26,12
3800 ? FIL$(ADES+2,BDES-1)
3810 IF FILE1$(3,10)=
     FIL$(ADES+2,BDES-5) THEN 3890
3820 ADES=ADES+15:BDES=BDES+15
3830 IF ADES>LEN(FIL$) THEN 3850
3840 GOTO 3710
3850 AFIL=AFIL+15:BFIL=BFIL+15
3860 GOTO 3700
3870 IF BFIL<LEN(FIL$) THEN 3710
3880 GOTO 1120
3890 C=1
3900 X=3
3910 POSITION 14,12
3920 ? "Rename Dups"
3930 IF FILE1$(X,X)=" " THEN GOTO 3980
3940 IF FILE1$(X,X)="." THEN X=X-1:GOTO 3980
3950 X=X+1
3960 IF X>11 THEN FIL$(AFIL,AFIL)="-":
     GOTO 3850
3970 GOTO 3930
3980 FILE1$(X,X)=STR$(C)
3990 ADES=1:BDES=15
4000 IF AFIL=ADES THEN 4060
4010 GOSUB 2020
4020 POSITION 1,12
4030 ? FIL$(ADES+2,BDES-1)
4040 IF FILE1$(3,10)=
     FIL$(ADES+2,BDES-5) AND C>9
     THEN FIL$(AFIL,AFIL)="-":GOTO 3850
4050 IF FILE1$(3,10)=FIL$(ADES+2,BDES-5)
     AND C<10 THEN C=C+1:GOTO 3980
4060 ADES=ADES+15:BDES=BDES+15
4070 IF BDES>LEN(FIL$) THEN GOTO 4090
4080 GOTO 4000
4090 FILE2$=FIL$(AFIL+2,BFIL-1)
4100 FIL$(AFIL,BFIL)=FILE1$
4110 FILE1$=FIL$(AFIL+2,BFIL-1)
4120 FOR X=1 TO LEN(FILE1$)-1
4130 IF FILE1$(X,X)=" " THEN
     FILE1$(X)=FILE1$(X+1):GOTO 4120
4140 GOSUB 2020
4150 NEXT X
4160 FOR X=1 TO LEN(FILE2$)-1
4170 IF FILE2$(X,X)=" " THEN
     FILE2$(X)=FILE2$(X+1):GOTO 4160
4180 GOSUB 2020
4190 NEXT X
4200 BF$(LEN(BF$)+1)="RENAME "
4210 BF$(LEN(BF$)+1)=FILE2$
4220 BF$(LEN(BF$)+1)=" "
4230 BF$(LEN(BF$)+1)=FILE1$
4240 BF$(LEN(BF$)+1)=CHR$(155)
4250 GOTO 3850
4260 POKE 559,34
4270 GOSUB 1990
4280 ? "There Are Too Many Files In ";
     ? PATH1$(19,21)
4290 ? "Hit [Return]->";
4300 GET #3,X
4310 GOSUB 1990
4320 GOSUB 4340
4330 GOTO 1810
4340 IF SC=0 THEN POKE 559,0
4350 RETURN
```

****

(18)

CHRISTIAN SOFTWARE

by Christian Software Developing Co.

Reviewed by Greg Payne, STarbase Associate

Christian Software, of Wabasha MN, has developed a "bible on disk" program, to be used with the Findex Data/Text Storage & Retrieval System program. (The Findex program is sold separately for $49.95.) I can't tell you if thay have other products out, as I've not seen their name around prior to this.

As you load the program and start to work with the data, it suddenly dawns on you that these disks are done in the original King James style. With all the modern day bible translations available, this surprised me. There are fewer and fewer readers & churches that use the original translations. The exception is the church pastor or serious bible student who needs to use the King James version, as it is the closest to the Greek & Hebrew manusrcipts. Anyway, once up & running, I noticed how similar it works to a bible concordance. All the books of the bible, old & new testament, were included.

The combination of the two programs, makes for an easy way to find a particular scripture, as long as you know some of the wording. If only it was translated, it would have been more enjoyable to work with. If you've got to have the bible on disk, then this programs for you.

If the Christian Software Devloping Co. proceeds with the next step & releases the program in modern day english, along with possibly having the ability to cross reference to the Greek & Hebrew versions, the program would be more fun to use. Until this happens, a good bible concordance is still more informative & simpler to use.

The Findex program was another matter. I found myself spending more time exploring & discovering ways to use it. Maybe another review is in order?

****

8-BIT PUBLIC DOMAIN PRODUCTIVITY (continued...)

command. In addition, you may change the format or width of a cell, column of cells, or the entire spreadsheet. As in SpeedScript you can change the text luminance and the background color. SpeedCalc has a MACRO capability allowing you to manipulate large or small amounts of data.

At this point, I need to apologize to Lonny who asked if this program could automatically update the cells when moving or copying them from one location to another. I had told him no but the answer is an emphatic yes. This type of move is called a *relative* move/copy as opposed to a *verbatim* move/copy.

The program has three Disk Commands; **CTRL D**, brings up the disk directory; **CTRL S**, shows SAVE: on the Command line and requests a valid file name including D#:; **CTRL L** (load), again the program prompts you to enter a valid file name. When the SAVE or LOAD functions are complete SpeedCalc lets you know there were not any errors. Should something go wrong it cancels the execution so as not to destroy or crash the program.

You may print to three different devices; to the screen for preview (E:); to a printer (P:); or to a Disk file (ASCII file) for integration of data with SpeedScript (D:filename). The print command is **CTRL P**, which then requests the "Device:" so you would then type E:, D:filename, or P:. Remember SpeedScript can only load files that have been PRINTED to a disk, not SAVED.

I think you'll like this program once you master it's very small amount of Commands.

ENJOY

****